Towards Robust Online Sexism Detection: A Multi-Model Approach with BERT, XLM-RoBERTa, and DistilBERT for EXIST 2023 Tasks

Hadi Mohammadi[1,*], Anastasia Giachanou[1] and Ayoub Bagheri[1].

[1]*Department of Methodology and Statistics, Utrecht University, The Netherlands.*

Abstract

This research investigates the application of pre-trained transformer-based models, including BERT, XLM- RoBERTa, and DistilBERT, in the context of the EXIST 2023 shared task, which focuses on identifying and categorizing online sexism. The study emphasizes the crucial role of Natural Language Processing (NLP) in detecting harmful content, and it draws on previous competitions that have incorporated tasks to detect hate speech and abusive language. The methodology combines various advanced techniques from the text classification domain, including the use of additional datasets, data preprocessing, and model building. The research also explores data augmentation techniques and label encoding as preprocessing steps. The study's findings indicate that the developed model performs optimally in English, and it suggests that the use of a voting system and the combination of outputs from multiple models contribute to the overall performance. The research concludes with a call for sustained initiatives to curb the prevalence of harmful content on digital platforms, and it outlines future work directions, including incorporating additional information about annotators, the assessment of annotator reliability, and exploring more sophisticated techniques for handling imbalances.

Keywords

Online Sexism, Natural Language Processing (NLP), Transformer-based Models, BERT.

1. Introduction

The advent of social media has revolutionized the way we communicate, interact, and disseminate information. However, this digital revolution that allows everyone to publish posts quickly and easily has resulted in a concerning amount of harmful content, such as hate speech, discrimination, and sexism. Despite efforts to mitigate these issues, the identification and categorization of sexism in online platforms remain challenging due to such expressions' nuanced and context-dependent nature [1]

Natural Language Processing (NLP) has witnessed remarkable progress in the past decade, primarily due to the emergence of deep learning algorithms for text classification [2]. These advancements have resulted in the development of models that can better understand and interpret human language, thereby paving the way for many applications, including identifying and classifying sexist content. Advanced Natural Language Processing (NLP) methods and Language Models can comprehend the subtleties and context-dependent language often employed in sexist expressions and, therefore, can be used to detect harmful content [3].

Several shared tasks and competitions, such as SemEval, have incorporated tasks to detect hate speech and abusive language. In 2023, the EXIST (sEXism Identification in Social neTworks) lab was

[1]CLEF 2023: Conference and Labs of the Evaluation Forum, September 18–21, 2023, Thessaloniki, Greece

EMAIL: email1h.mohammadi@uu.nl (H. Mohammadi); a.giachanou@uu.nl (A. Giachanou); a.bagheri@uu.nl (A. Bagheri)
ORCID: 0000-0003-0860-9200 (H. Mohammadi)

launched as part of the CLEF (Conference and Labs of the Evaluation Forum) conference focusing on addressing the problem of sexism detection [4]. To address those challenges, we participated in the EXIST 2023 competition on sexism detection in social media platforms [5]. EXIST 2023 provides a dataset of over 10,000 labeled tweets in English and Spanish which comprises three tasks: Identification of sexism, source intention, and categorization of sexism. Sexism identification is a binary classification task that requires systems to decide whether a tweet contains or describes sexist expressions or behaviors. Source intention is a categorization task that classifies the intention behind the sexist messages into one of three pre-defined categories: direct, reported, or judgmental. The final task, sexism categorization, is a multiclass and multilabel task that categorizes the sexist messages according to the type(s) of sexism they contain. Those categories, such as ideological and inequality, stereotyping and dominance, objectification, and sexual violence, after considering the undermined aspects of women [6].

In the remainder of this paper, we present our methodology for tackling the EXIST 2023 tasks in Section 2, we discuss our experimental results in Section 3, and finally, in Section 4, we summarize our work and present future work directions.

Methodology

We have developed a methodology combining various advanced techniques from the text classification domain to address the sexism detection task. Our methodology consists of several steps, which are shown in Figure 1:

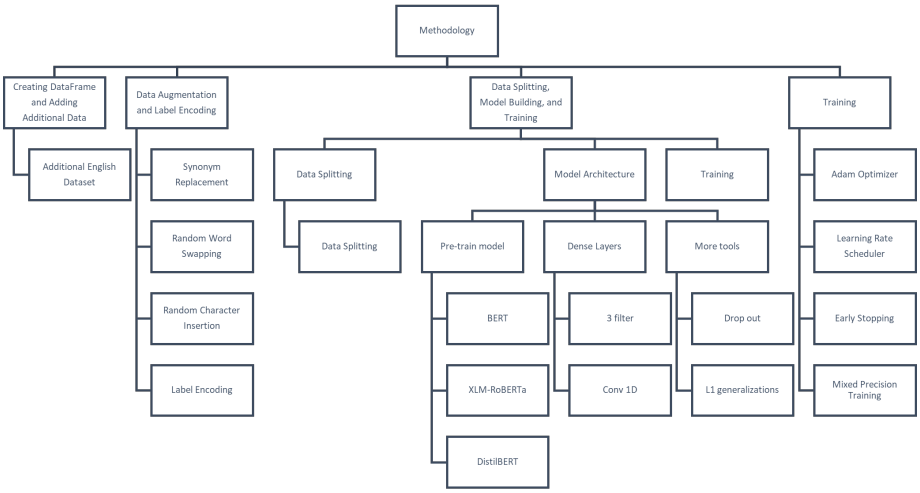

Figure 1: Model Architecture Visualization

1.1. Additional Dataset Description

In addition to the EXIST 2023 dataset, we also utilized another valuable resource for our research: the dataset provided by the SemEval-2023 Task 10: Explainable Detection of Online Sexism (EDOS); this dataset was introduced to address the issue of binary detection of sexism, which often overlooks the diversity of sexist content and fails to provide clear explanations for why something is considered sexist [7]. This dataset contains 20,000 social media comments with fine-grained labels. By integrating the EDOS dataset into our methodology, we enhanced the robustness and explainability of our models for online sexism detection.

1.2. Model Building

The model-building phase of this research is a distinctive element of the methodology, with a unique architecture that incorporates three different transformer-based models (BERT, XLM- RoBERTa, DistilBERT). Transformer-based models are a category of models that use self-attention mechanisms and have achieved state-of-the-art results in various natural language processing tasks.

BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based model pre-trained on a large corpus of text data. BERT considers the full context of a word by looking at the words that come before and after it, which is why it is described as bidirectional. Understanding the context from both directions allows BERT better to understand the semantic meaning of words and sentences.

As our second model, we used XLM-Roberta, a variant of the RoBERTa model (a robustly optimized version of BERT), which is trained on a large amount of multilingual data making it highly effective for multilingual tasks.

DistilBERT is a lighter version of BERT, created through a process called distillation, where the knowledge of the larger BERT model is transferred to a smaller, faster model. Despite being smaller and faster, DistilBERT maintains most of the performance of BERT, making it a good choice for tasks where computational resources are limited.

Each transformer model processes the input text independently, generating a unique data representation. These representations are then concatenated and passed through a Conv1D layer. Following the Conv1D layer, a MaxPooling1D layer is used. After the pooling layer, the data is flattened. Flattening is a simple operation that transforms a multi-dimensional array into a one-dimensional array. This step is necessary because the output of the MaxPooling1D layer is a 2D tensor, while the subsequent layers require a 1D tensor. The flattened data is then subjected to a Dropout layer. Finally, two Dense layers are used. Dense layers are the regular deeply connected neural network layers. In this case, one Dense layer is used for binary classification (task1), and the other for multiclass classification (task2 and 3). These layers output the final predictions of the model, which can then be evaluated and used for making decisions.

2. Experimental Setup

In this section, we describe the experimental setup we followed to assess the performance of our models. This setup encompassed a range of processes, from system configuration and hyperparameter optimization to model training and validation, all aimed at ensuring the most accurate and reliable results.

2.1. Data Preprocessing

Data loading and preprocessing are critical initial phases in machine learning or data analysis tasks. This process prepares the raw data for subsequent steps, such as model training and testing. After we loaded the data, we continued with the preprocessing phase. The first step in preprocessing was to convert all textual content to lowercase. This step is essential for normalizing the data and reducing complexity, as machine learning algorithms treat uppercase and lowercase versions of the same character as different entities. By converting all text to lowercase, we can ensure that the same word in different cases is recognized as the same word by the algorithm. Next, we removed elements that contained no information for this task, including URLs, memorable characters, and punctuation. Removing them helps reduce the data's dimensionality and makes extracting meaningful information easier for the algorithm. Following this, the text was tokenized and lemmatized. Tokenization is the process of dividing text into individual words or tokens. This step is necessary because machine learning algorithms work with individual units (like words) rather than entire texts. Lemmatization is a further refinement of this process. It reduces words to their base or root form, grouping different grammatical forms of the same word. This helps reduce the data's complexity and makes it easier for the algorithm to identify patterns. Finally, we incorporated the EDOS dataset to enrich the data pool. We followed the same preprocessing steps. Incorporating additional data can help improve the model's robustness and increase its generalizability to unseen data.

Data augmentation is a strategy used to increase the diversity and amount of training data without collecting new data. This technique is primarily used to prevent overfitting, a common problem where a model performs well on training data but poorly on unseen data. By creating modified versions of the existing data, the model can learn more robust features and generalize better to new data.

In this research, we used the following data augmentation techniques from the" nlpaug" library in Python:

- **Synonym Replacement:** This technique involves replacing words in the text with their synonyms. This increases the linguistic diversity of the dataset and helps the model to understand that different words can have the same meaning.
- **Random Word Swapping:** This technique involves randomly swapping pairs of words in the text. This introduces a controlled noise level into the data and helps the model learn that the order of words can vary while preserving the overall meaning.
- **Random Character Insertion:** This technique involves randomly inserting characters into words. This also introduces a controlled noise level and helps the model learn to handle typos or other minor errors in the input data.

Label encoding is a preprocessing step that converts categorical labels into a format that machine learning algorithms can use. Most algorithms expect numerical input and output, so it is necessary to transform categorical labels into numerical values. In this research, we used a majority voting system to decide the labels for the sexism detection task (task 1). This means that the label assigned to each data point was the one that was chosen by the majority of annotators. These labels were binary (indicating the presence or absence of a particular feature) and were encoded into a binary format for compatibility with the machine learning model. For tasks 2 and 3, a multiclass label encoding process was used. This process involved encoding each class as a unique integer. This is a standard method for handling multiclass problems, where each data point can belong to one of several classes. A variety of machine-learning algorithms can easily use the resulting numerical labels.

2.2.Model Training

In the training phase, the model learns to map inputs (features) to outputs (labels) based on the training data. We used the Adam optimizer for training. Adam, short for Adaptive Moment Estimation, is a popular choice for deep learning applications due to its adaptive learning rates, meaning it adjusts the learning rate for each weight in the model individually. A learning rate scheduler from callbacks in TensorFlow with a learning rate of 3e-05 and warmup steps of 200 was incorporated to adjust the learning rate during training dynamically. This helps to fine-tune the learning process, often leading to better model performance. An early stopping mechanism was also utilized to prevent needless training once the model's performance ceased to improve significantly. This not only saves computational resources but also helps prevent overfitting. Mixed precision training was employed to expedite the training process. This method involves using a mix of single-precision (float32) and half-precision(float16) data types during training, which can significantly reduce the use of computational resources without compromising the model's performance.

2.3.Different Runs

In We conducted three different runs for Task 1, which involved Sexism Identification. Each run represented a unique combination of methods and data, allowing us to explore various strategies for improving the performance of our model.

- **Task 1 - Run 1 (Binary label- Original dataset):** In the first run, we trained our model Without using additional data. We used a voting system that combined the predictions of several models (BERT, XLM-RoBERTa, and DistilBERT) to make a final decision. This approach, often called ensemble learning, can help improve the robustness and accuracy of predictions by leveraging the strengths of multiple models. The task was treated as a binary classification problem, with the model predicting whether each instance was sexist.
- **Task 1 - Run 2 (Binary label- Original and additional dataset):** In the second run, we included additional features derived from the text, metadata, and external resources (EDOS). Like the first run, we used a voting system with several models and treated the task as a binary classification problem. The additional information enriched the feature space and gave the model more context for making predictions.

- **Task 1 - Run 3 (Multi-label- Original and additional dataset):** In the third run, we again incorporated additional information and used a voting system with several models. However, we treated the task as a multilabel classification problem in this run. This means that each instance could belong to more than one class. We had two labels for the original data (indicating whether the instance was sexist) and two based on the additional data. This approach allowed us to capture more nuanced information about the instances and potentially improve the model's ability to detect different forms or levels of sexism.

Our methodology for tasks 2 and 3 is like task 1- run 3. Through different runs, we explored a range of strategies for sexism identification and gained insights into their relative strengths and weaknesses. These findings can inform future work in this area, helping to guide the development of more effective tools for detecting and combating online sexism.

2.4. Evaluation and Validation

Model performance is generally defined by its ability to predict unseen data accurately. One of the most popular of these techniques is cross-validation. Cross-validation aims to estimate the model's accuracy on the test set during the model training stage. In this regard, we divide the original data into k partitions of equal size. One partition is designated a validation dataset, and the remaining k-1 partitions are used as training data. These stages enable iterative enhancements to the model and its training process, ultimately creating a reliable machine-learning model that can effectively generalize to new, unseen data.

2.4.1. System Configuration

Our system leverages a suite of transformers for text processing sourced from the Hugging Face Transformers library, a state-of-the-art platform for natural language processing. After

preprocessing, we set a max length of 256 for tokenization, reflecting the average length of the text data. The num-labels parameter was set to 1, indicating binary classification. Hyper-parameters were optimized via Random Search within a defined range of possible values. We examined learning rate values between 1e-5 and 1e-4 and assessed batch sizes 16, 32, and 64 to determine the optimal balance between computational efficiency and model performance. We also implemented a learning rate scheduler that dynamically adjusts the learning rate according to a cosine decay schedule. The warmup period was set to 200 steps, with incremental steps computed based on the number of epochs and the dataset size. We utilized early stopping based on validation loss to prevent overfitting, with the patience set to 3 epochs.

2.4.2. Model Training and Validation Process

The models were trained using binary cross-entropy loss and the Adam optimizer, a popular choice for deep learning applications due to its adaptive learning rates. We utilized the mixed-precision training policy 'mixed float16' to expedite training without compromising model performance. We employed a custom function to construct models, with each model utilizing a different transformer: $bert-base-multilingual-uncased$, $xlm-roberta-base$, and $distilbert-base-multilingual-cased$. Outputs from these transformers were concatenated and passed through a $1D$ convolutional layer, a max-pooling layer, a dropout layer (rate of 0.5), and a dense layer for binary classification with $L2$ regularization.

Validation was carried out via stratified dataset splitting, with the test size constituting 20 percent of the entire data. This data was tokenized and passed through the trained models. The model training process involved searching for the optimal set of hyperparameters for 3 trials using the validation set. A final model was trained and evaluated after identifying the best hyperparameters (learning rate and batch size).

2.5. Evaluation and Validation

To gain a comprehensive understanding of our results, we visualized the data distribution and the proportion of the labels in our data using plots like bar charts, pie charts, or histograms. This provided valuable insights into the imbalance of the dataset. The distribution of task one is given in the diagram below; due to the large volume of task two and three labels, we avoided bringing its diagram.

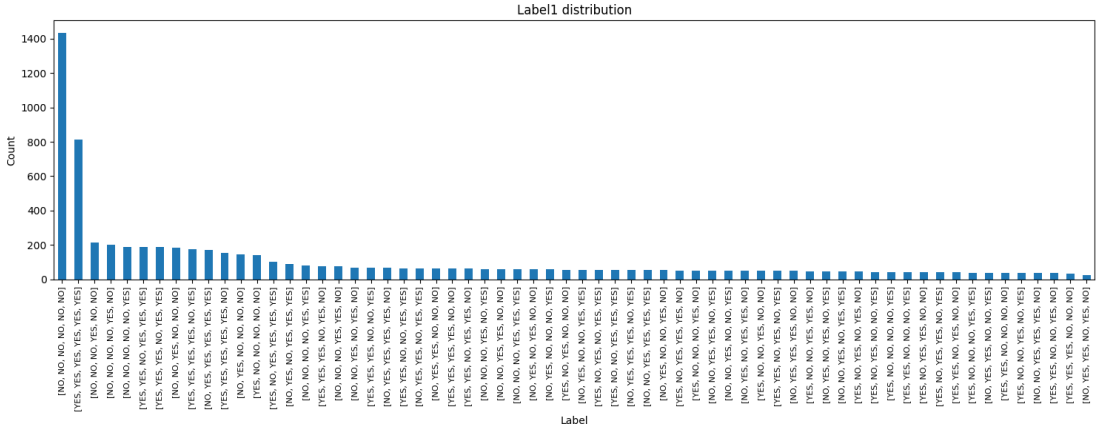

Figure 2: label 1 distribution

We also conducted a text length analysis to examine the lengths of the text data before and after preprocessing. This helped us determine an appropriate max-length value for the tokenization and padding step.

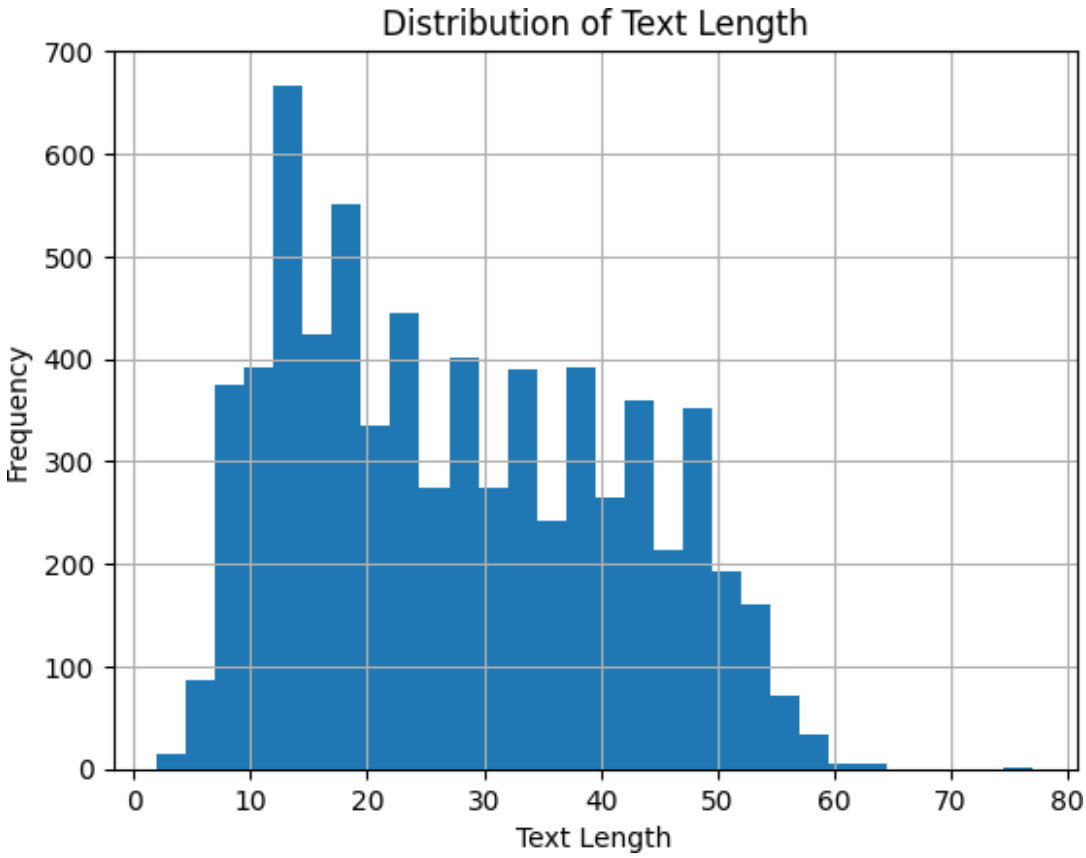

Figure 3: Distribution of Text Length

We plotted the loss and accuracy progress over the epochs during the training phase. This provided insight into whether the model was learning effectively or overfitting/underfitting. After training, we evaluated our model on the test data and generated a classification report and a confusion matrix.

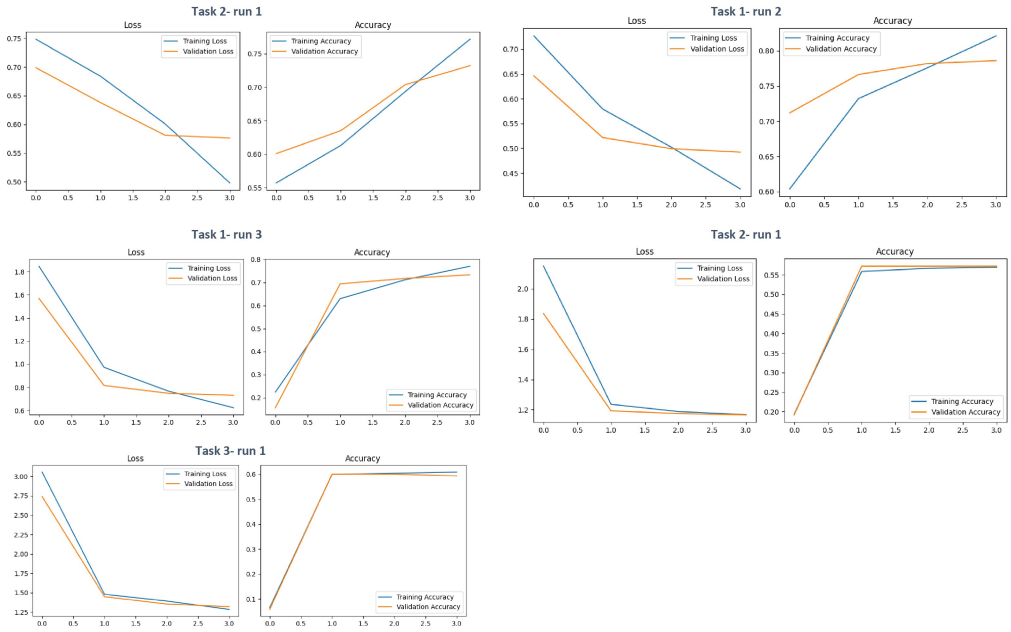

Figure 4: Training Performance

Despite using advanced and similar pre-trained models, various methods, like the early stopping method, have prevented overfitting. The performance of our model on the test

dataset was evaluated based on several key metrics: accuracy, precision, recall, and F1 score. In the below table, performance metrics for each task in the test dataset have been shown.

Table 1
Model Performance metrics for each task run on the test dataset

Task	Accuracy	Precision	Recall	F1 score
Task 1- run 1	61.03%	37.25%	61.03%	46.26%
Task 1- run 2	78.06%	64.49%	54.96%	59.34%
Task 1- run 3	67.38%	61.90%	67.38%	64.19%
Task 2- run 1	57.56%	33.13%	57.56%	42.06%
Task 3- run 1	60.02%	48.47%	60.02%	52.32%

According to table 2, in task one, it is clear that the best accuracy was obtained in run two, where the total data of the original and additional data were used. The classification was considered in binary form, indicating that the model has been more accurate on the total data; however, the F index has decreased due to the additional data set in English. Also, the best F1 score is obtained in the third run, related to using the total data and multi-labeling (two labels for the original data and two for the additional data).

In our model's evaluation, we have utilized Accuracy, Precision, Recall, and F1-score. How- ever, the evaluation competition adopted metrics such as ICM-Hard, ICM-Soft, their normalized versions, and Cross-Entropy, designed to evaluate the model's performance under various scenarios, including hard and soft classifications. The 'hard' scenarios deal with discrete, categorical classifications, similar to Accuracy in our current framework, while the 'soft' scenarios handle continuous, probabilistic classifications. Majority and minority class classifiers were also used as baselines to compare the models' performance over a naïve approach. Our team's best run was ranked **3rd** among the participants which was related to $Task\ 3\ Hard-Soft\ EN$. The details of the official result are shown in the below table.

Table 2
Official competition results for the task 3

EXIST_2023_Leaderboard_Task3					
Task 3 Soft-Soft ALL	Run	Rank	ICM-Soft	ICM-Soft Norm	
	M&S_NLP_1	**14**	-8,3574	0,6793	
Task 3 Hard-Hard ALL	Run	Rank	ICM-Hard	ICM-Hard Norm	F1
	M&S_NLP_1	31	-2,1587	0,1838	0,0017
Task 3 Hard-Soft ALL	Run	Rank	ICM-Soft	ICM-Soft Norm	
	M&S_NLP_1	**4**	-9,504	0,6586	
Task 3 Soft-Soft ES	Run	Rank	ICM-Soft	ICM-Soft Norm	
	M&S_NLP_1	11	-8,5493	0,6701	
Task 3 Hard-Hard ES	Run	Rank	ICM-Hard	ICM-Hard Norm	F1
	M&S_NLP_1	30	-2,2525	0,192	0,0026
Task 3 Hard-Soft ES	Run	Rank	ICM-Soft	ICM-Soft Norm	
	M&S_NLP_1	**6**	-9,6746	0,6496	
Task 3 Soft-Soft EN	Run	Rank	ICM-Soft	ICM-Soft Norm	
	M&S_NLP_1	11	-7,939	0,6957	
Task 3 Hard-Hard EN	Run	Rank	ICM-Hard	ICM-Hard Norm	F1

	M&S_NLP_1	30	-2,0384	0,179	0,0007
Task 3 Hard-Soft EN	Run	Rank	ICM-Soft	ICM-Soft Norm	
	M&S_NLP_1	**3**	-9,1251	0,6745	

The official results, along with a thorough soft and hard analysis, reveal that the developed model performs optimally in English. This outcome may be attributed to the use of a multilingual model as opposed to a Spanish-focused one, as well as the relative abundance of English language information in these models. The use of additional datasets, which were exclusively in English, could also have influenced this result. Future endeavors include experimentation with multiple models catered to different languages. It's also noteworthy to highlight that the developed model outperformed competing teams in the multi-class classification problem (the third task). This success is likely due to the implementation of a voting system and the combination of outputs from multiple models, which collectively contributed to the overall performance.

3. Conclusions and Future Work

Our research presents a comprehensive approach to online sexism detection, leveraging advanced machine learning techniques and natural language understanding methodologies. We utilized a multi-model approach incorporating BERT, XLM-RoBERTa, and DistilBERT to tackle this complex issue. Our methodology was evaluated through a robust experimental setup, and the results demonstrated the effectiveness of our approach in both data sets, but it needs to be improved in dealing with unbalanced datasets by considering the lack of sexist text in the original and additional datasets.

Looking ahead, we plan to extend our research by incorporating additional information about the annotators, such as gender, age, and other demographic details. This information could

provide valuable context that may influence the interpretation and categorization of sexist content. For instance, research has shown that perceptions of sexism can vary significantly based on an individual's personal experiences and perspectives (Burn, 2000; Swim et al., 2005) Moreover, we aim to assess the reliability of the annotators. Annotator reliability is a crucial aspect of any study involving human annotation, as it can significantly impact the quality and validity of the data [8]. By evaluating the reliability of our annotators, we can ensure that our dataset is robust and reliable, thereby enhancing the validity of our findings. In addition, we plan to explore more sophisticated techniques for handling imbalanced data and methods for dealing with subtlety and context dependency in sexist expressions. We also aim to test our model on different datasets and contexts to assess its generalizability.

3.1. Model Training

In the training phase, the model learns to map inputs (features) to outputs (labels) based on the training data. We used the Adam optimizer for training. Adam, short for Adaptive Moment Estimation, is a popular choice for deep learning applications due to its adaptive learning rates, meaning it adjusts the learning rate for each weight in the model individually. A learning rate scheduler from callbacks in TensorFlow with a learning rate of 3e-05 and warmup steps of 200 was incorporated to adjust the learning rate during training dynamically. This helps to fine-tune the learning process, often leading to better model performance. An early stopping mechanism was also utilized to prevent needless training once the model's performance ceased to improve significantly. This not only saves computational resources but also helps prevent overfitting. Mixed precision training was employed to expedite the training process. This method involves using a mix of single-precision (float32) and half-precision(float16) data types during training,

which can significantly reduce the use of computational resources without compromising the model's performance.

We plotted the loss and accuracy progress over the epochs during the training phase. This provided insight into whether the model was learning effectively or overfitting/underfitting. After training, we evaluated our model on the test data and generated a classification report and a confusion matrix.

4. Data Availability

In this article, the dataset utilized is specifically associated with EXIST 2023 competition. Furthermore, the code utilized in the study was made accessible through a dedicated Zenodo [10].

5. References

[1] Z. Waseem, D. Hovy, Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter, in: Proceedings of the NAACL Student Research Workshop, Association for Computational Linguistics, San Diego, California, 2016, pp. 88–93. URL: https://aclanthology.org/N16-2013. doi:10.18653/v1/N16-2013.

[2] T. Young, D. Hazarika, S. Poria, E. Cambria, Recent Trends in Deep Learning Based Natural Language Processing, 2018. URL: http://arxiv.org/abs/1708.02709. doi:10.48550/arXiv.1708. 02709, arXiv:1708.02709 [cs].

[3] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, A. T. Kalai, Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings, in: Advances in Neural Information Processing Systems, volume 29, Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/hash/a486cd07e4ac3d270571622f4f316ec5-Abstract.html.

[4] J. Pavlopoulos, P. Malakasiotis, I. Androutsopoulos, Deeper Attention to Abusive User Content Moderation, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 1125–1135. URL: https://aclanthology.org/D17-1117. doi:10.18653/v1/ D17-1117.

[5] L. Plaza, J. Carrillo-de Albornoz, R. Morante, E. Amigó, J. Gonzalo, D. Spina, P. Rosso, Overview of EXIST 2023: sEXism Identification in Social NeTworks, in: Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III, Springer-Verlag, Berlin, Heidelberg, 2023, pp. 593–599. URL: https://doi.org/10.1007/978-3-031-28241-6_68. doi:10.1007/ 978-3-031-28241-6_68.

[6] L. Plaza, J. Carrillo-de Albornoz, R. Morante, J. Gonzalo, E. Amigó, D. Spina, P. Rosso, Overview of exist 2023: sexism identification in social networks, in: Proceedings of ECIR'23, 2023, pp. 593–599. doi:10.1007/978-3-031-28241-6_68.

[7] H. R. Kirk, W. Yin, B. Vidgen, P. Röttger, SemEval-2023 Task 10: Explainable Detection of Online Sexism, 2023. URL: http://arxiv.org/abs/2303.04222. doi:10.48550/arXiv.2303.04222, arXiv:2303.04222 [cs].

[8] R. Artstein, M. Poesio, Survey Article: Inter-Coder Agreement for Computational Linguis-

[9] tics, Computational Linguistics 34 (2008) 555–596. URL: https://aclanthology.org/J08-4004. doi:10.1162/coli.07-034-R2.

[10] H. Mohammadi, A. Giachanou, & A. Bagheri. (2023). Code for "Towards Robust Online Sexism Detection: A Multi-Model Approach with BERT, XLM-RoBERTa, and DistilBERT for EXIST 2023 Tasks". URL: https://doi.org/10.5281/zenodo.8144300.

6. Appendix

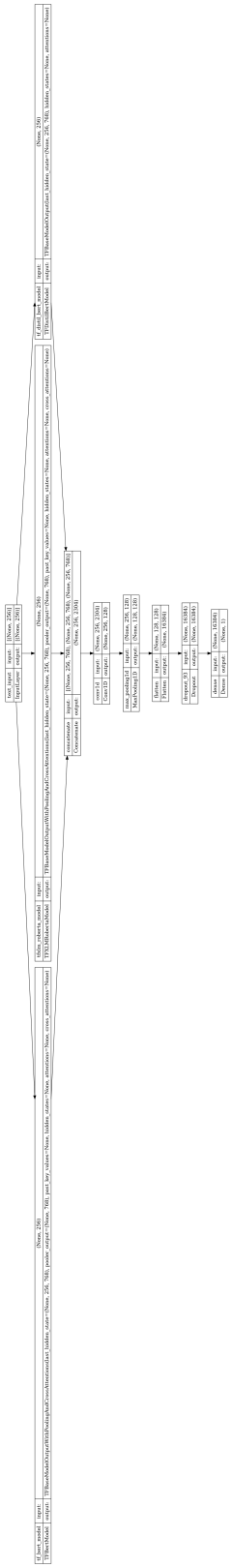

Figure 2: Model Architecture Visualization

To gain a deeper insight into the model utilized, refer to the diagram above which illustrates its structure. The outputs from three pre-trained language models are compiled by a single CNN model,

which then categorizes the information based on the task at hand. Detailed information about the number of parameters can be found in the table provided below.

Table 1
Model Summary

```
Model: "model"
_____________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==============================================================================================
 text_input (InputLayer)        [(None, 256)]        0           []

 tf_bert_model (TFBertModel)    TFBaseModelOutputWi  167356416   ['text_input[0][0]']
                                thPoolingAndCrossAt
                                tentions(last_hidde
                                n_state=(None, 256,
                                 768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 tfxlm_roberta_model (TFXLMRobe  TFBaseModelOutputWi  278043648  ['text_input[0][0]']
 rtaModel)                      thPoolingAndCrossAt
                                tentions(last_hidde
                                n_state=(None, 256,
                                 768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 tf_distil_bert_model (TFDistil  TFBaseModelOutput(l  134734080  ['text_input[0][0]']
 BertModel)                     ast_hidden_state=(N
                                one, 256, 768),
                                 hidden_states=None
                                , attentions=None)

 concatenate (Concatenate)      (None, 256, 2304)    0           ['tf_bert_model[0][0]',

'tfxlm_roberta_model[0][0]',

'tf_distil_bert_model[0][0]']

 conv1d (Conv1D)                (None, 256, 128)     884864      ['concatenate[0][0]']

 max_pooling1d (MaxPooling1D)   (None, 128, 128)     0           ['conv1d[0][0]']

 flatten (Flatten)              (None, 16384)        0           ['max_pooling1d[0][0]']

 dropout_93 (Dropout)           (None, 16384)        0           ['flatten[0][0]']

 dense (Dense)                  (None, 1)            16385       ['dropout_93[0][0]']

==============================================================================================
Total params: 581,035,393
Trainable params: 581,035,393
Non-trainable params: 0
```